

1. 如果有 3 个异常处理 `hi` 过程: `catch(...)`、`catch(const void *)`、`catch(int *)`, 应如何摆放它们的位置? `catch(int *)` 可以放在 `catch(const void *)` 后面而不影响其捕获异常吗?

答案: (1) 根据下面的顺序摆放: `catch(int *)`、`catch(const void *)`、`catch(...)`。
(2) 将 `catch(int *)` 放在 `catch(const void *)` 后面, 将导致 `catch(int *)` 失效。

2. `int x`, 显式转换 `(int)x` 的结果是左值还是右值?

答案: 右值。

3. 对于全局变量 `const int x = 0`, 能够使用 `*const_cast<int *>(&x) = 3` 修改 `x` 的值吗? 用 `*(int *)(&x) = 3` 可以修改 `x` 的值吗?

答案: 都不能。非类内定义的简单类型变量即使转换为左值, 也不能修改其值 (受到页面保护机制的保护)。

4. 说明 `static_cast` 和 `const_cast` 的区别。

答案: `static_cast` 和 `const_cast` 同 C 语言的强制类型转换用法基本相同, 但 `static_cast` 不能修改源类型中的 `const` 和 `volatile` 属性, `const_cast` 则可以。

5. 对于 Lambda 表达式: `auto f = [x] (int y) -> int { return ++x + y; }`, 解释 `[x] (int y)` 中的 `x` 和 `y` 的意义, 并说明为什么 `f` 名义上是一个函数, 但实际上是一个对象。

答案: `[x]` 表示捕捉 lambda 函数体外变量 `x` 的值, 并根据这个值创建匿名类的实例成员变量 `x`。`(int y)` 表示匿名类的 `()` 运算符的重载函数 `operator()(...)` 的调用参数, 即 `int operator()(int y)`。

6. 类模板的实例化有哪几种方式?

答案: 2 种方式: 显式实例化和隐式实例化。

例如, 对于类模板 `MAT<T>`,

显式实例化: `template class A<1>;` (等价 `template class A<int>;`)

隐式实例化: `MAT<int> a(2,3);` (实例化类模板 `MAT` 并且创建对象 `a`)

7. 分析下面的程序, 指出变量 `g1 ~ g4`、`a1 ~ a6` 中, 哪些变量的地址是相同的 (指向同一个对象)? 执行完指令 “`float &a5 = f<float>();`” 和 “`float a6 = f<float>();`” 后, 各变量的值是多少?

```
template<typename T, int x=0>
```

```
T g = T(10 + x);
```

```
template float g<float>;
```

```
template<typename T>
```

```

T &f() {
    T &a = g<T, 0>;
    return ++a;
}

float g1 = g<float>;
float &g2 = g<float>;
const float &g3 = g<float>;
const float &g4 = g<float, 4>;

int main()
{
    float a1 = g<float>;
    float &a2 = g<float>;
    const float &a3 = g<float>;
    const float &a4 = g<float, sizeof(float)>;
    float &a5 = f<float>();
    float a6 = f<float>();
}

```

答案：g2、g3、a2、a3、a5 指向同一个对象（显式创建的、初始值为 10.0f 的匿名变量）。

g4、a4 指向同一个对象（隐式创建的、初始值为 14.0f 的匿名变量）。

g1、a1、a6 分别指向不同的存储单元。

执行完 “float &a5 = f<float>()”：a1=g1=10, g4=a4=14, g2=g3=a2=a3=a5=11

执行完 “float a6 = f<float>()”：a1=g1=10, g4=a4=14, g2=g3=a2=a3=a5=a6=12

分析：

“template float g<float>” 显式地创建全局 float 类型的匿名变量（值为 10.0f）。

“float g1 = g<float>;” 创建变量 g1，将前面创建的值为 10.0f 的匿名变量的值拷贝到 a1。

“float &g2 = g<float>;” 引用变量 g2 引用前面创建的值为 10.0f 的匿名变量。

“const float &g3 = g<float>;” 引用变量 g3 引用前面创建的值为 10.0f 的匿名变量。

“const float &g4 = g<float, 4>;” 隐式创建全局 float 类型的匿名变量（值为 14.0f）。

“float a1 = g<float>;” 创建变量 a1，将前面创建的值为 10.0f 的匿名变量的值拷贝到 a1。

“float &a2 = g<float>;” 引用变量 a2 引用前面创建的值为 10.0f 的匿名变量。

“const float &a3 = g<float>;” 引用变量 a3 引用前面创建的值为 10.0f 的匿名变量。

“const float &a4 = g<float, sizeof(float)>;” 引用变量 a4 引用前面创建的值为 14.0f 的匿名变量。

“float &a5 = f<float>();” f()返回前面创建的值为 10.0f 的匿名变量的引用，并将该匿名变量的值加 1，引用变量 a5 引用该匿名变量。

“float a6 = f<float>();” f()返回前面创建的初始值为 10.0f 的匿名变量的引用，并将该匿名变量的值加 1，并将该值拷贝到新的变量 a6。

- 设计一维数组模板 ARRAY，可以实例化各种简单类型的一维数组。在模板的成员函数中需要考虑抛出异常（如内存不足、数组下标越界等），并在 main()中进行简单的测试（包括捕获异常处理）。

答案：参考实验四。

9. 设计二维数组模板 `ARRAY2`，可以实例化各种简单类型的二维数组。模板中包含一个非类型形参，表示构造函数的一个参数的缺省值，用于初始化数组元素的值。并给出任意一条实例化模板的语句。

答案：参考实验四。